
# Embedded Visual Processing Platform 2001

*Neil Trevett*
*Senior Vice President, Market Development*

# What is Embedded Visual Processing?

*An emerging market opportunity for graphics vendors*

- **Embedded = any platform that is not a traditional PC or workstation**
  - Covers a broad range of appliances, devices and vehicles
- **Visual Processing = any mix of 2D and 3D graphics, video and imaging**
  - Processing pixels
  - Breaking down traditional barriers between generated graphics acquired imagery
- **90% of broadband bandwidth will be used for visual information**
  - According to Scientific American
- **Visual processing – will be an essential element of many appliances**
  - What are the opportunities and challenges?

# Topics Explored by this Presentation

## *Embedded visual processing markets and technologies*

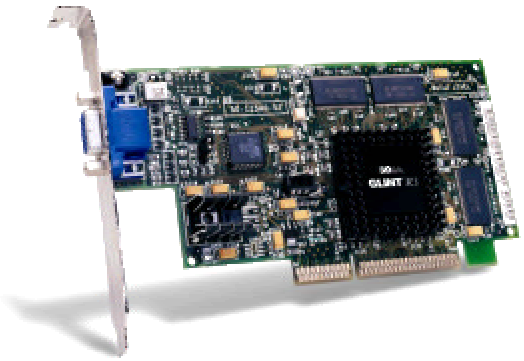
- **3Dlabs and our recent focus on embedded visual processing**
- **A taxonomy of the embedded visual processing market**
  - Analyzing the diverse needs for advanced visual processing
- **The need for open standard APIs to drive standardized functionality**
  - And how to get involved in a new initiative defining these APIs
- **Architectural Issues for embedded graphics**
  - Is tile-based rendering the way to go?
- **IP cores for SoC designs will be essential for appliances**
  - Market swinging from fabless semiconductor vendors to IP vendors
  - Challenges and first steps

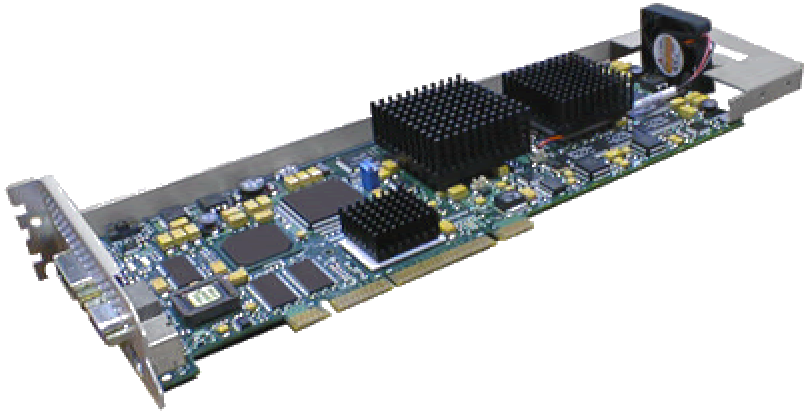
3Dlabs

# Unique Graphics Expertise

*Generating graphics IP is our core competency*

- **We have been selling visual processors for almost 20 years**
  - Unique competence in 2D graphics, 3D graphics, imaging and video
- **One of the most highly skilled graphics centers of excellence**
  - Chip, board and software expertise with over 100 visual processing patents
- **Multiple chip, board and independent chip design teams**
  - Over 250 people total worldwide – including over 150 engineers
- **Worldwide Operations**
  - Three main locations – Huntsville, Silicon Valley and London
  - Global Sales and support – including sales offices in Germany and Japan

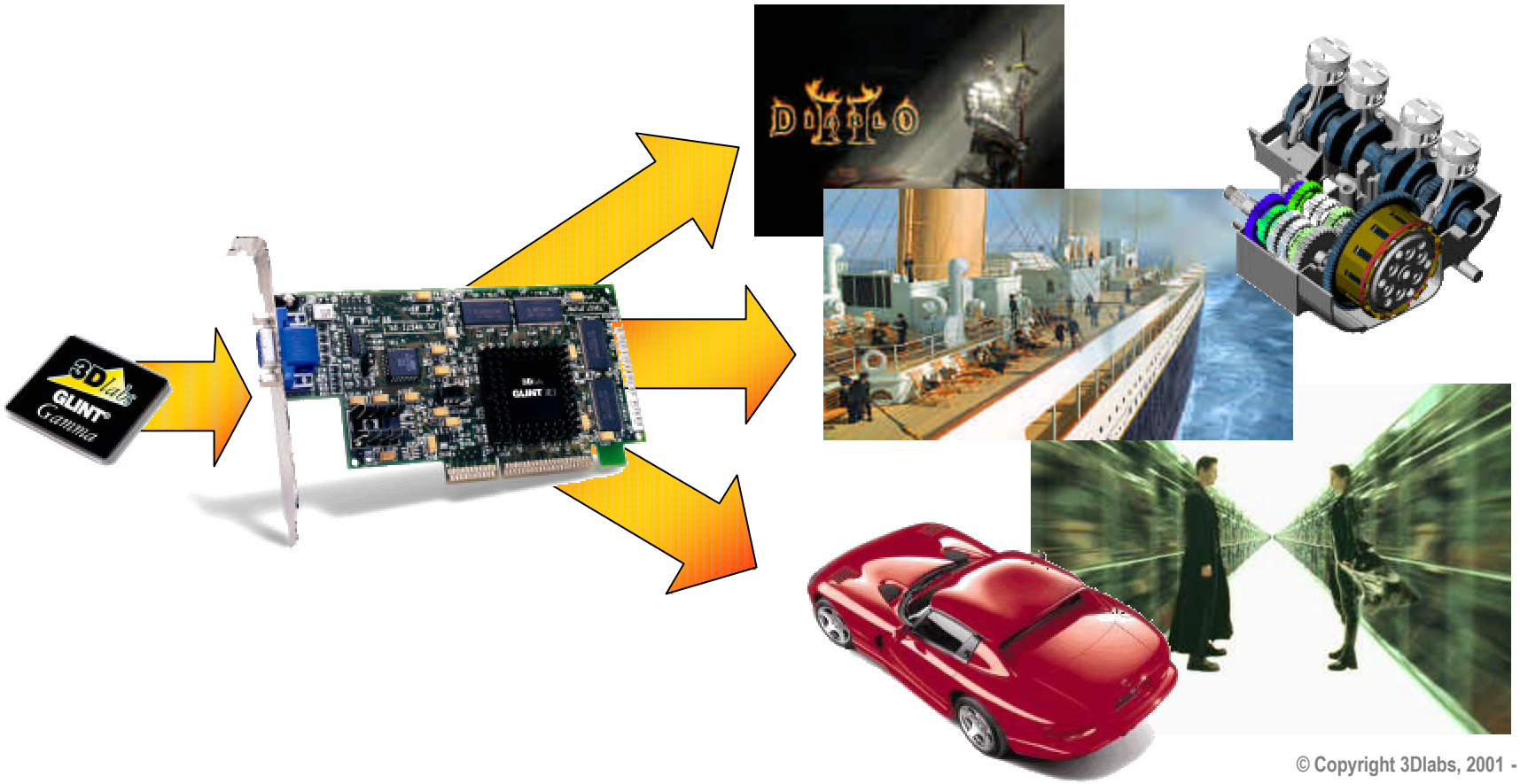




# 3Dlabs' Traditional Business

## *Market Leaders in Professional Graphics*

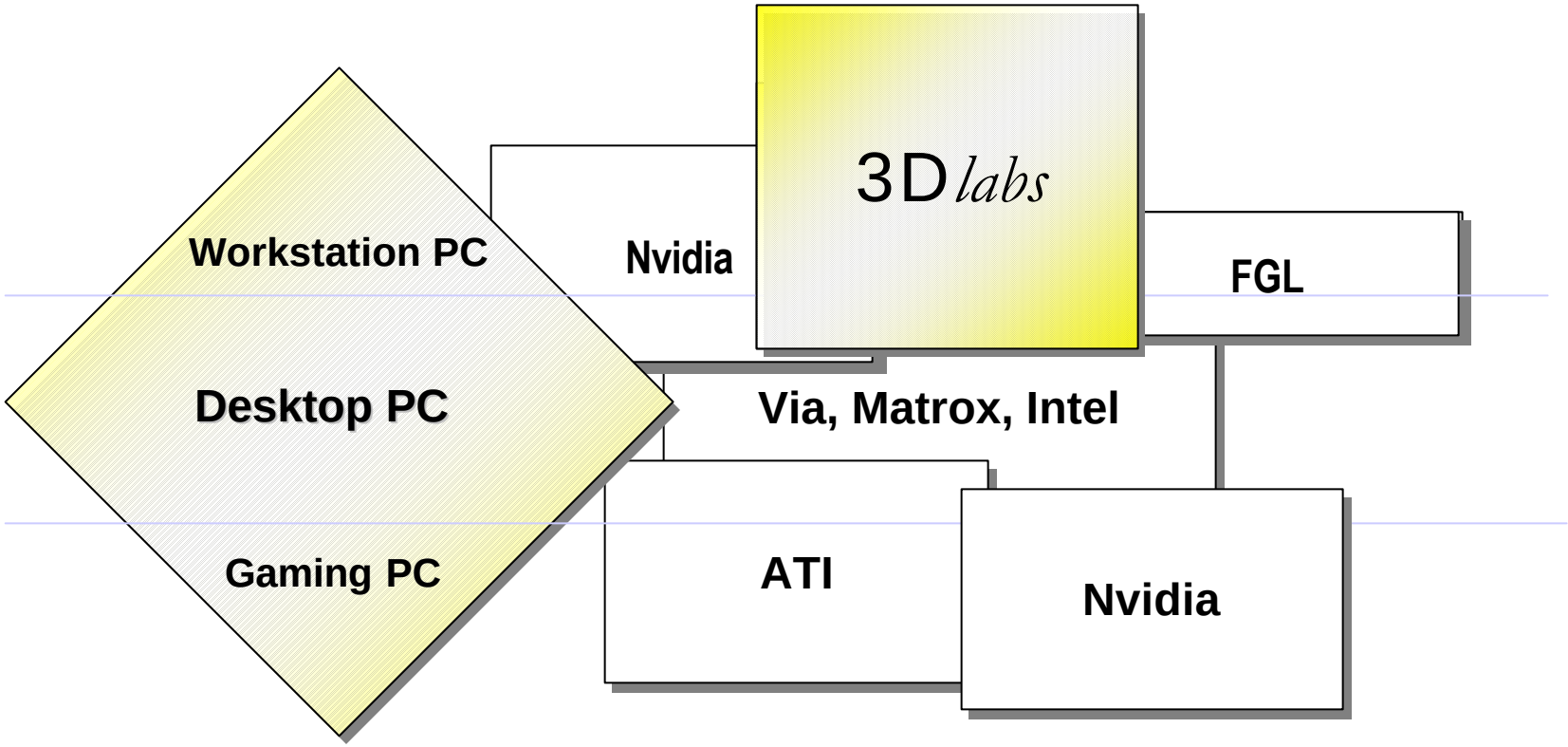
- **We sell 3D graphics accelerators to CAD and Digital Media Designers**
  - Boosting graphics design productivity through stunning 3D quality and performance
- **60% market share for professional 3D graphics**
  - Our products are used to create everything from web-sites through feature films to airliners



# 3Dlabs' Dominant Market Position

*Leading professional graphics industry consolidation*

- E&S have withdrawn from the workstation board market
- FGL has no internal IP
- Nvidia are focused on games
- The acquisition of DPI and Intense3D by 3Dlabs has created a clear market leader



# 3Dlabs' Professional Graphics Channel

## *A strategic strength*

- **3Dlabs has the industry's largest professional graphics channel**
  - A robust 50/50 split between OEM and VAR business
- **Every Tier One OEM in the worlds uses 3Dlabs product**
- **Worldwide distribution channel**
  - Over 50 distributors in 42 countries serving VARS and system integrators
  - Over 740 qualified VARS

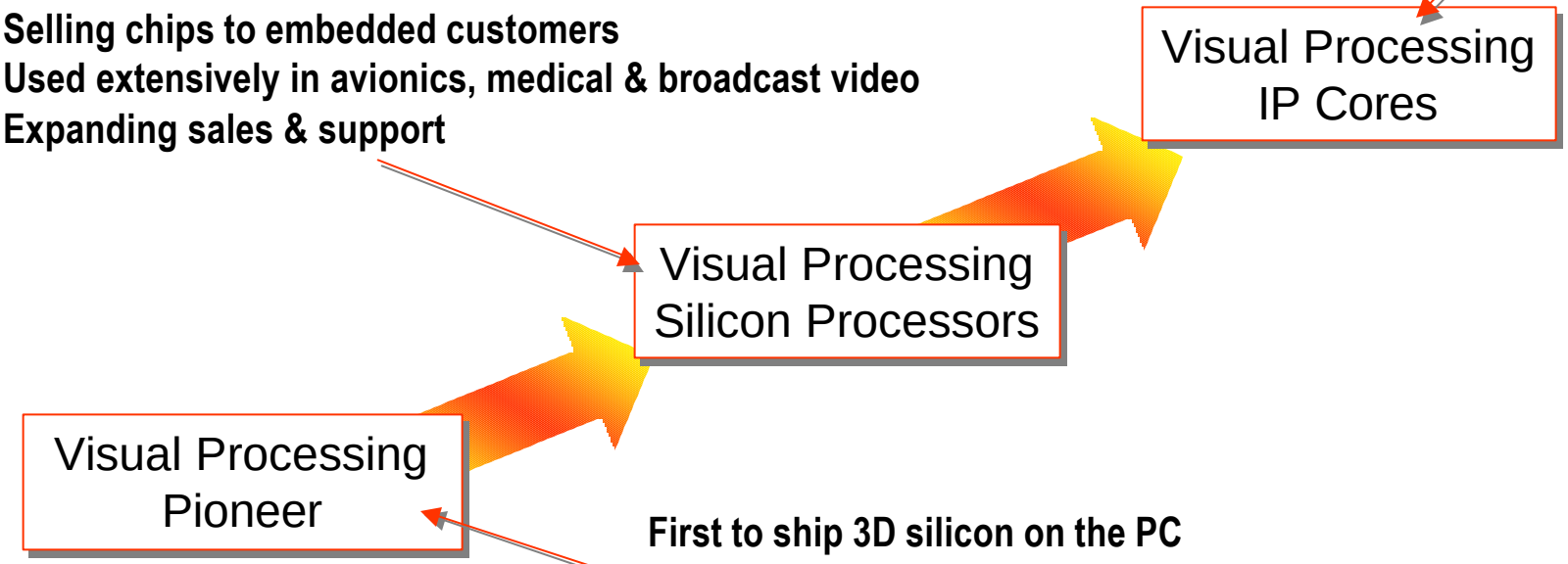
# 3Dlabs and Embedded Graphics

## *An increasing focus and commitment*

- **3Dlabs is the leading supplier of professional graphics processors**
- **We intend to be the #1 embedded visual processing supplier**
  - The same emphasis on quality and advanced functionality

**Now offering intellectual property for SOC designs**
**High-volume, low-cost applications**
**Automotive, STB, internet appliances**

**Selling chips to embedded customers**
**Used extensively in avionics, medical & broadcast video**
**Expanding sales & support**




**First to ship 3D silicon on the PC**
**Market leader in workstation graphics**
**The fastest graphics on the planet**

# 3Dlabs Chip Market Success

## *Our silicon used widely in the embedded industry*

- **We have significant embedded chip sales**
  - In avionics, professional video and medical markets
- **Significant customer base in systems market**
  - Bae, Barco, Dayang, Dy4, Eaton, Elbit, Ericsson, Electrosonic, Imagine Graphics,
  - Intraserver, Imco, Kaiser, Lockheed, Pinnacle, Radstone, Sony, TechSource, VDO-L
- **Our processors are unmatched in the embedded graphics market**
  - Advanced 2D and 3D functionality
  - High quality rendering
  - The industry's most robust OpenGL
  - Accelerated OpenGL on VxWorks
  - Unparalleled service and support
  - Strong driver partnerships

The F22 flies with GLINT GMX avionics display subsystems developed by Kaiser Electronic

# 3Dlabs Embedded Graphics Division

*Focused on the needs of embedded OEMs*

**Well defined support program**
Dedicated support staff

**Long term supply program**
Solving the problem of using rapid generation parts in long deployment programs

**Packaged Visual IP**
Easy to use IP for SoC designs with visual processing


**Timely access to silicon**
All 3Dlabs processors offered to embedded customers without delay

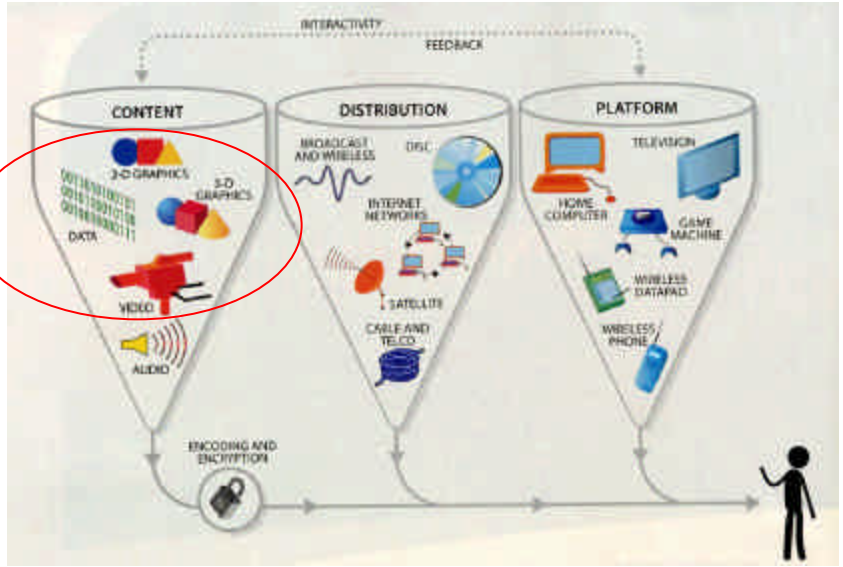
**Visual API Leadership**
Leading the creation of new embedded visual processing API standards

**Extensive RTOS Support**
Including VxWorks
Full 2D/3D graphics acceleration

# Market Functional Requirements

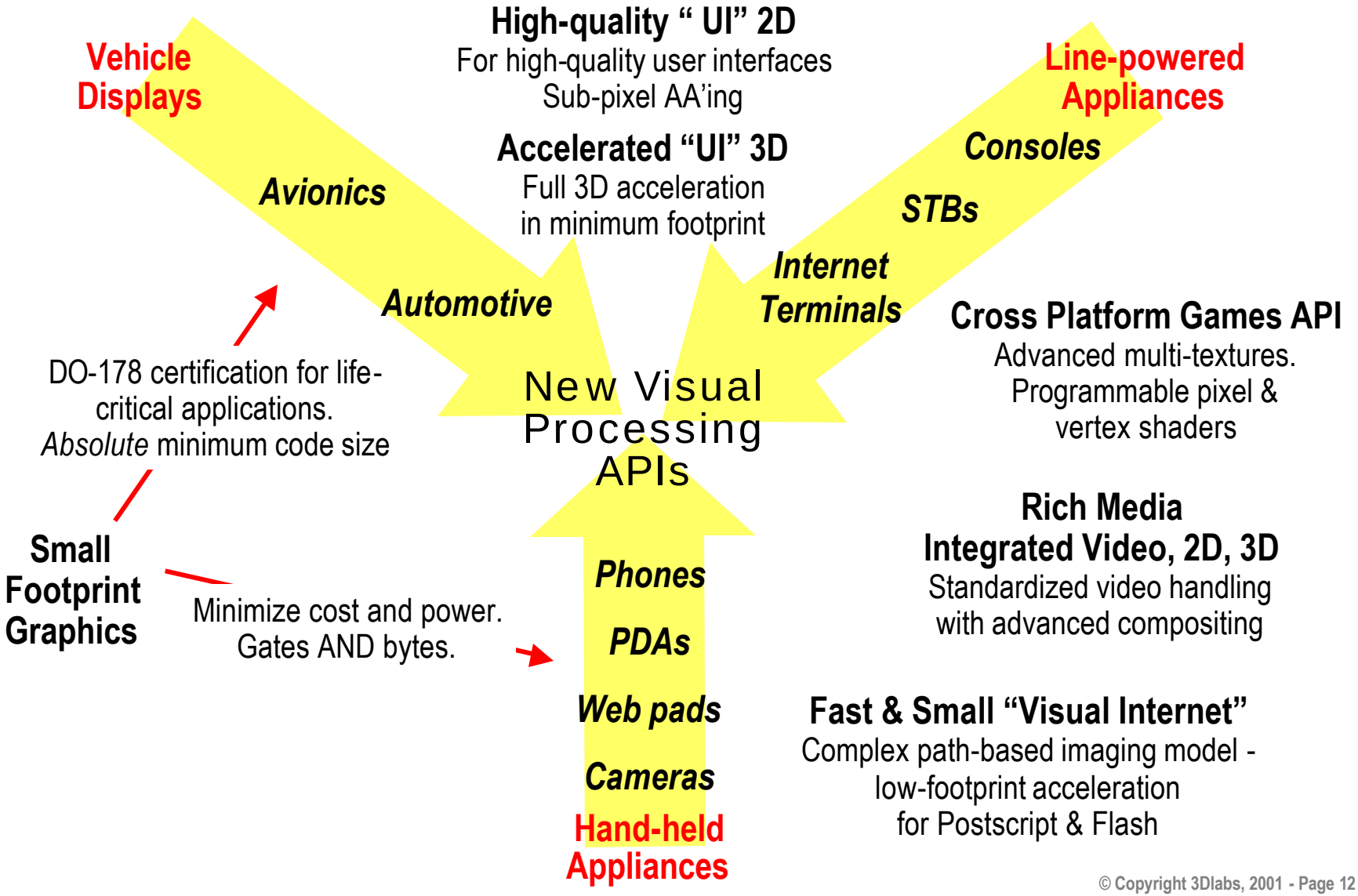
## *Through defining embedded visual processing APIs*

- **Embedded devices and requirements are becoming better defined**
  - The graphics community should be evangelizing for advanced visual capabilities
- **The graphics hardware industry is fundamentally changing**
  - Delivering acceleration as tunable, flexible IP cores rather than monolithic chips
- **This enables finely-tuned graphics acceleration for specific markets**
  - Tremendous flexibility – also tremendous danger of graphics market fragmentation
- **Open Visual APIs are the only way to avoid market chaos**
  - A clean "synch point" for agreed functionality between hardware and software communities
  - Enable widespread market growth
- **3Dlabs is leading an embedded open standards initiative**
  - To create open visual processing API standards



# Three Requirements "Vectors"

## *Converging on a need for Open, OS-agile Visual APIs*



# Cross Platform Solution Essential

*Many platforms and initiatives need these APIs*

**WindML® – expanding multimedia capabilities of VxWorks®**
Looking to further expand 3D and rich media delivery capabilities

**Embedded Linux – need for low-footprint APIs**
No standardized 2D API other than X Windows – not low footprint
Increasing use of OpenGL for 3D – but not low footprint

**Java Community – Java2D, Java Media Framework, Java Advanced Imaging**
Java3D is a high-level API – need low-footprint 3D API for embedded applications

**ARINC – standards for advanced cockpit displays**
Need standardized 2D & 3D APIs
Severe certification demands drive need for minimum API size

**EPOC – emerging OS for mobile consumer devices**
Needs minimal API size and increasing functionality

**Web3D Consortium**
Extensible standards for 3D delivery over the internet, driving 3D in MPEG-4
BUT needs target small footprint 2D/3D API for higher-level standards

# The Need for Advanced Graphics

## *In small-screen appliances*

- **The 'Inverse Screen-size Rule'**
  - "The smaller the screen, the more advanced the visual processing that is needed to effectively display information on it"
- **Opportunity to fundamentally differentiate appliances**
  - With the quality, usability and "industrial-design" of the user interface

**Complete 2D and 3D functionality to support a wide range of applications**

**Flexible memory architecture to flexibly handle one or more video streams**

**High quality filtered 3D for advanced user interface techniques**


**Translucent menus for high user friendliness without taking screen area**

**Anti-aliased text for readability at small font sizes**

# Current API Initiatives

*We need a holistic view of embedded requirements*

- **But existing groups are working mainly on PC-based graphics**
  - Could they be coordinated for the embedded market?

**OpenGL® ARB**
The proven industry-standard, cross-platform 2D/3D API
BUT too much workstation functionality for low footprint implementations

**Microsoft DirectX**
Excellent integrated rich-media delivery APIs
BUT only available for Windows

**BCL – Broadcast Compositing Language**
A surfacing API for rich media client playback – good foundation for cross-media graphics
Also looking at small footprint graphics – strong focus on settop box profile
BUT a relatively small number of participating companies

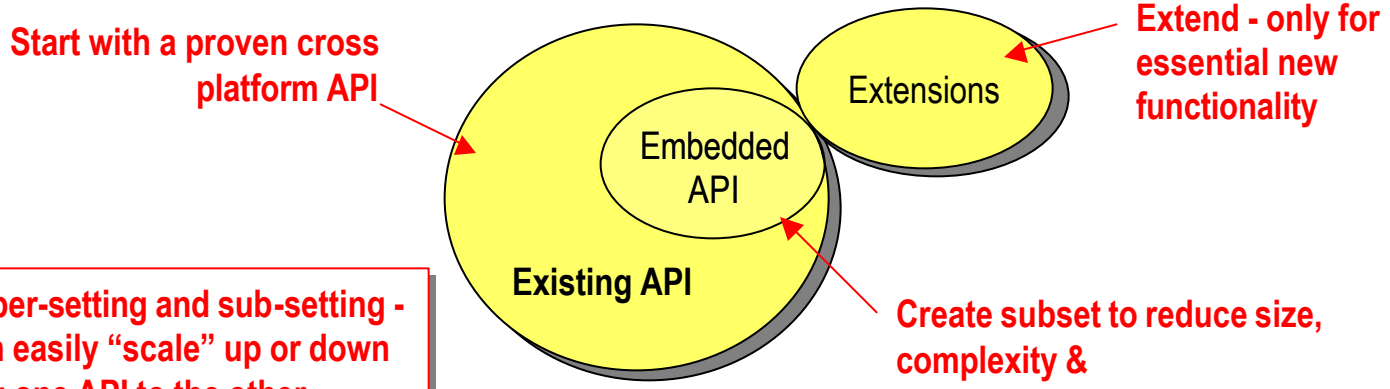
**OpenML**
OpenGL extensions for video, control of video devices for authoring
BUT no work yet on small footprint, client-side playback of rich media

# Proposed API Framework

## *Minimize effort and use existing momentum*

- **Work at lowest level that enables desired functionality**
  - The thinnest possible layer on top of the hardware – OpenGL-like
- **Adopt existing standards wherever possible**
  - Supporting open standards to enable market growth
- **Subset existing standards for low-footprint APIs**
  - Proven functionality and upward scalability to full APIs
- **Extend existing APIs for needed new functionality**
  - Enabling advanced digital media capabilities
- **Separate functionality into separate APIs**
  - Applications link only the APIs they need for configuration flexibility and minimum footprint

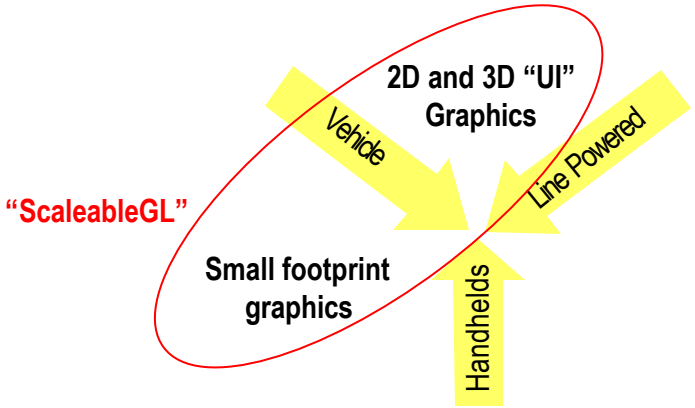



With a proper super-setting and sub-setting - applications can easily "scale" up or down from using one API to the other

# "ScalableGL" Graphics

*For creating embedded visual applications*

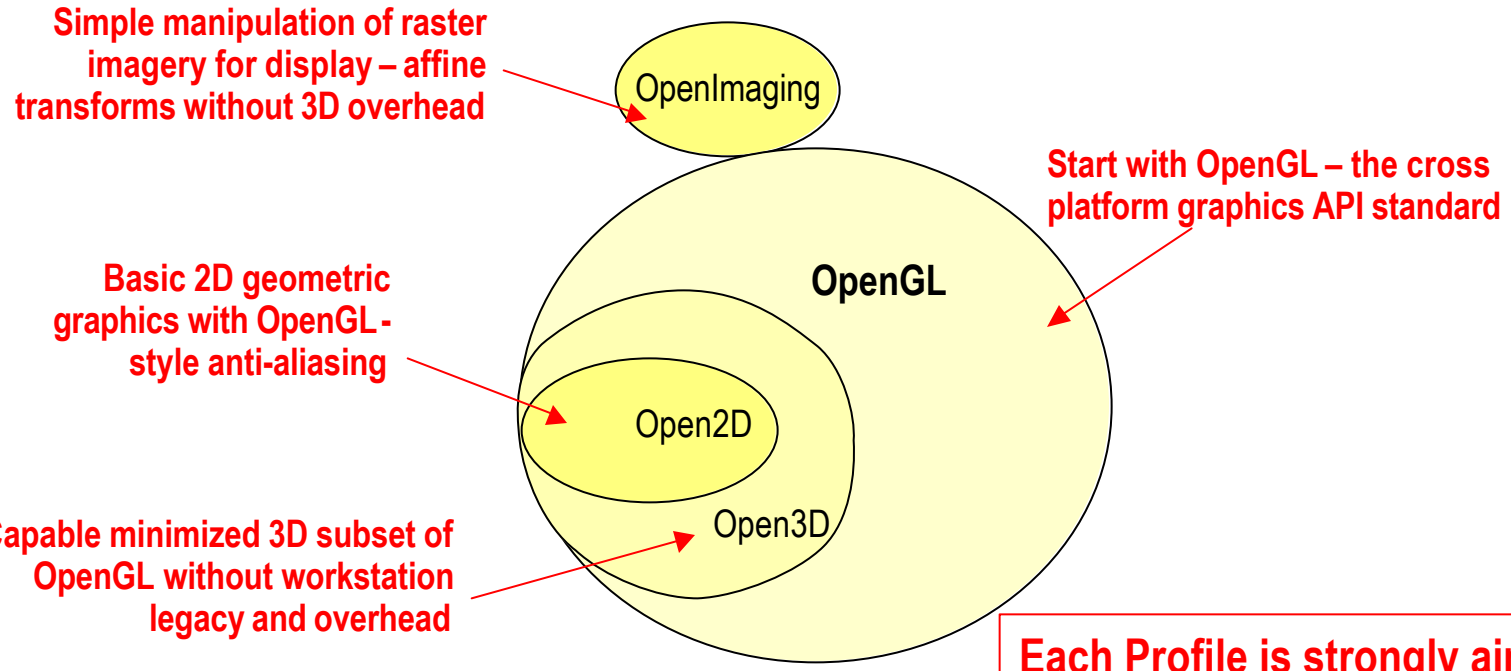
- **Issues with existing APIs**
  - 2D – there are no suitable existing APIs
    - Almost all are not cross platform (e.g. Wind River's UGL)
    - Cross platform 2D APIs tend to be bundled with complex windowing (e.g. X Windows)
    - Many are 70s/80's vintage and are not low-level – e.g. define arcs and ovals as primitives
    - None provide advanced sub-pixel anti-aliasing
  - Raster Image display – small footprint handling not possible in OpenGL
    - Mapping displays require simple image affine transforms – filtered rotate and zoom
    - Only way to rotate an image in OpenGL is to use the full 3D texture pipeline
  - 3D – OpenGL functionality is widely accepted – but has workstation overhead
    - No agreed OpenGL subsets – the industry is fragmenting – everyone inventing their own APIs
- **Proposed Solution – "ScaleableGL"**
  - A coherent set of OpenGL profiles to deliver robust 2D and 3D in minimum footprint
  - "Open2D", "Open3D" and "OpenImaging"



# Proposed "ScalableGL" APIs

## *Meeting application requirements*

- **A straightforward rationale for scaling OpenGL**
- **Rational, hardware-neutral scaling of large functional units**
  - Minimize quibbling over individual function calls
- **Eliminating redundancy, legacy and workstation functionality**
  - Needs discussion over exact subsets and functionality of each subset




**Each Profile is strongly aimed at a particular market segment**

# Application Example

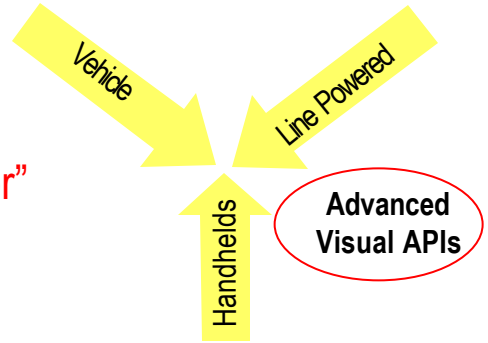
## *Proposed APIs - solution for Avionics Displays*

- **Cockpit Display - typical requirements**
  - Anti-aliased 2D geometric graphics
  - No windowing system
  - Minimum possible code size to minimize DO-178 certification costs
  - SOLUTION – Open2D
- **Enhanced raster requirements**
  - Display of raster maps
  - Rotation required for “direction of flight up” mode
  - SOLUTION – Open2D + OpenImaging
- **Enhanced 3D required**
  - For display of “out of window” 3D terrain
  - SOLUTION – Open3D + OpenImaging

# Advanced API Requirements

## *New capabilities for embedded visual processing*

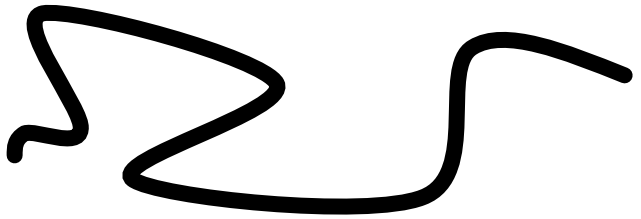
- **Video Handling**
  - Need a portable way of setting up and controlling video processing streams
  - Building blocks for digital media stream manipulation and processing
  - Solution – consider a client/playback subset of OpenML "ML" API – "OpenMedia"
- **Seamlessly integrate video with 2D & 3D graphics**
  - One solution is to define "surfaces"
  - Any "marking engine" API writes to a surface
  - Surfaces can be combined with advanced methods
  - Solution – consider using surface API from BCL
- **Gaming 3D**
  - Need an open API that matches and exceeds capabilities of Direct3D
  - Advanced multi-texturing and programmable vertex and pixel shaders
  - Solution – consider extensions to Open3D – "OpenGaming"
- **Vector-based 2D**
  - New imaging model to cleanly accelerate Postscript and Flash
  - High-quality 2D graphics with flawless sub-pixel anti-aliasing
  - Complex paths PLUS "squeegee" operations through the path
  - Solution – consider an embedded subset of SVG – "OpenVector"



# Vector Graphics

## *Learning from SVG by the W3C*

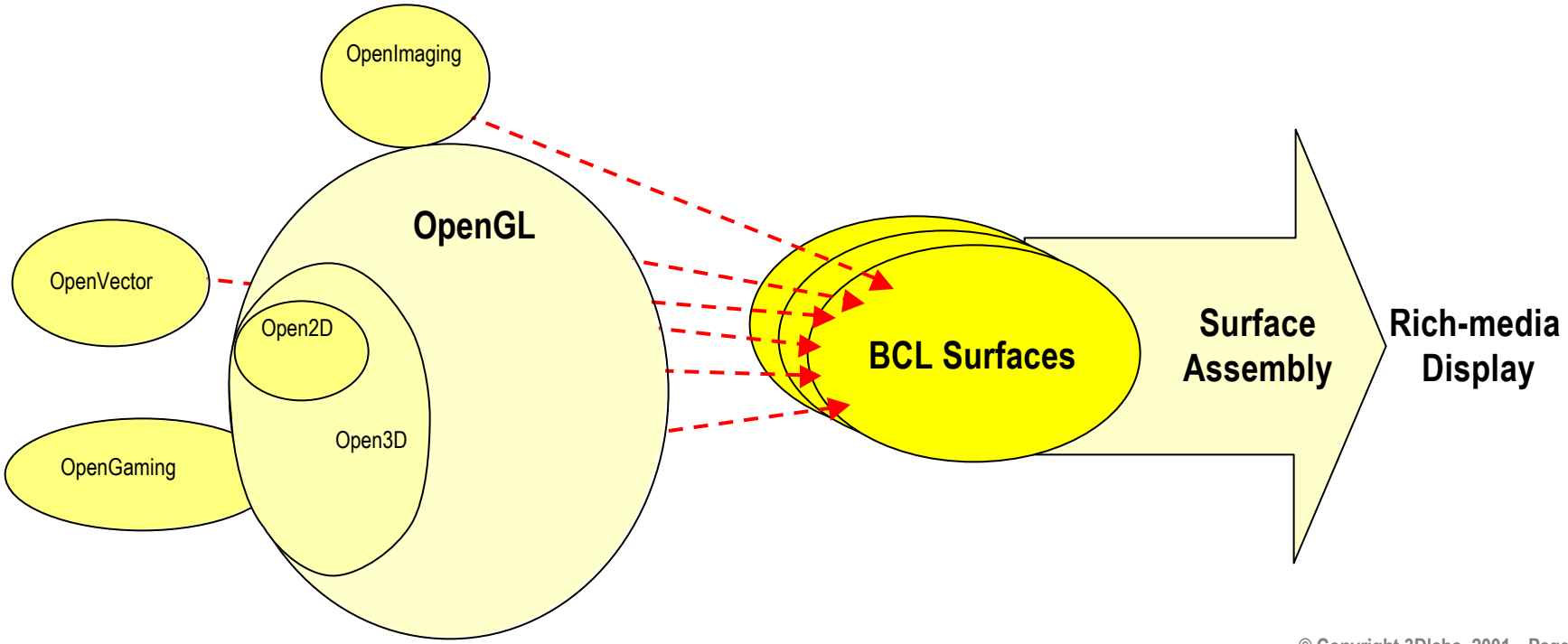
- **SVG by the W3C uses a very simple imaging model**
  - Define smooth curved path – then “squeegee” various operations through it
- **Ideal web graphics imaging model**
  - Native and natural acceleration for Flash/PostScript renderers
  - Backed by a formidable collection of companies and is already available from Adobe
  - Includes support for web fonts and custom fonts
- **Better suited than OpenGL for web graphics**
  - Support for cap styles and join styles - a big issue for real 2D applications
  - Robust support for text and fonts – OpenGL must use texture maps to get good quality
  - Efficient support for paths and squeegee fill and composite operations
- **OpenVector – a simple hardware-oriented API**
  - Enabling low-footprint, accelerated, advanced web graphics



# BCL Surfaces

*Unifying framework for all other graphics APIs*

- **BCL enables the creation and control of surfaces**
  - All graphics APIs are “marking engines” onto BCL surfaces
- **Surfaces can be combined using advanced methods**
  - Overlays, underlays, translucent compositing
  - Enables cross-media compositing
- **BCL Surface API is implemented independently of any marking API**



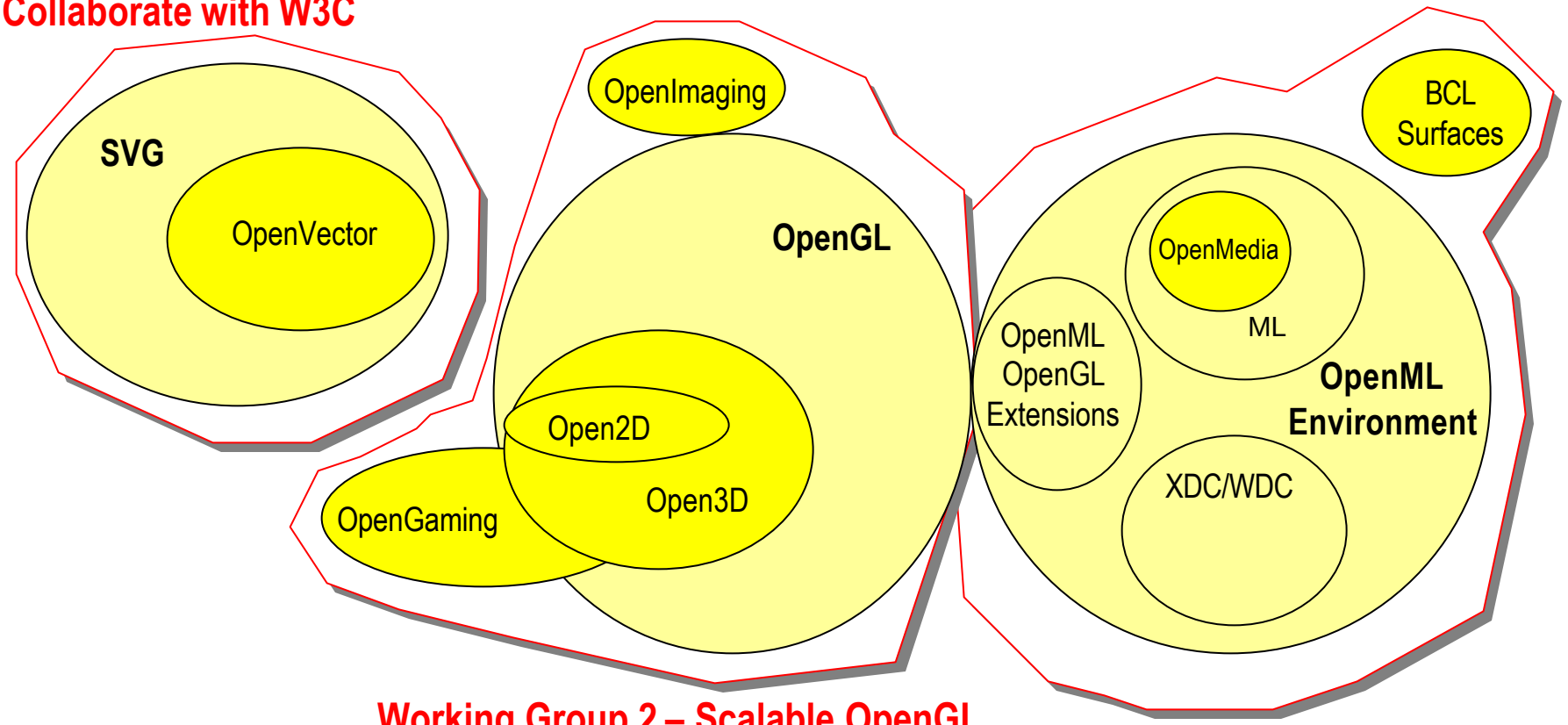
# Embedded Visual Processing APIs

## *A Comprehensive Framework*

- **Multiple points for focused effort and innovation**

**Working Group 1 – Vector Graphics**
**Collaborate with W3C**

**Working Group 3 – Media Display**
**Collaborate with BCL**




**Working Group 2 – Scalable OpenGL**
**Collaborate with ARB**

# 3Dlabs Leading this Initiative

## *Encouraging Khronos Group to take up this work*

- **Any group to effectively drive this initiative needs:**
  - A fast start – there is an immediate urgent need – fragmentation is happening already
  - An industry quorum – of both the graphics AND embedded industries
  - Open voting membership - embedded industry players require a vote on the spec
  - Open disclosure of work in progress – for embedded industry feedback
  - Should produce standards that are patent unencumbered and royalty free
  - Needs to address all embedded API needs – not just OpenGL-related ones
- **3Dlabs is active in all key standards organizations**
  - OpenGL ARB, Khronos/OpenML, Web3D Consortium,
  - BroadcastCL, GPC/OPC, ARINC
- **Need involvement by users and generators of embedded technology**
  - If you are interested to get involved contact neil.trevett@3dlabs.com



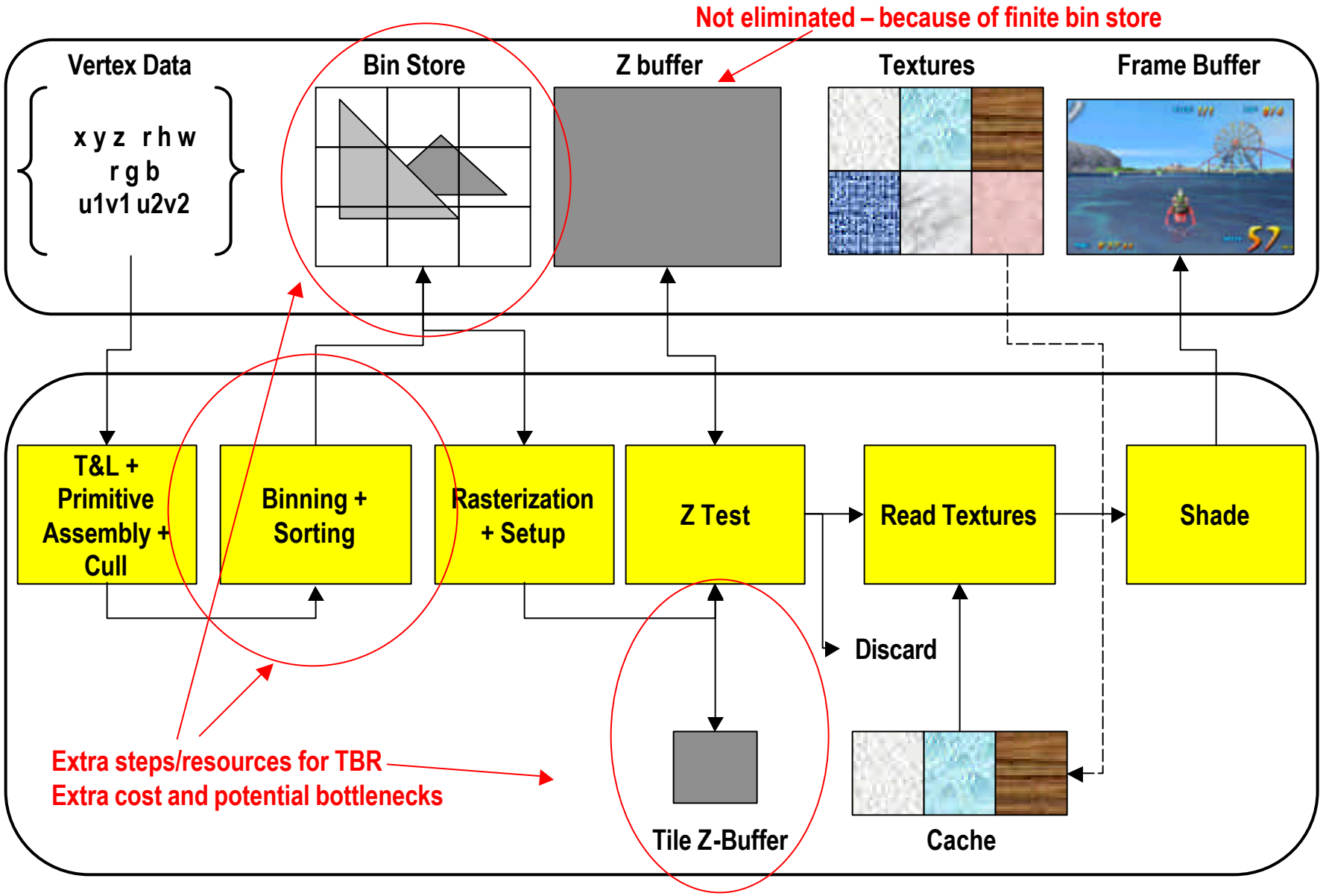


# Tile-Based Rendering

*Is it the right way to implement embedded graphics?*

- **Tile-based rendering has potential advantages for small appliances**
  - Binning – eliminate Z reads/writes from/to external Z buffer – eliminate Z-buffer memory
  - Sorting - eliminates overwrites – increasing performance per gate
- **However – there are hidden costs and disadvantages**
  - And ways for traditional architectures to gain many of the advantages of TBR
- **Lets analyze and compare the traditional and TBR pipelines…**

# Comparing the two pipelines

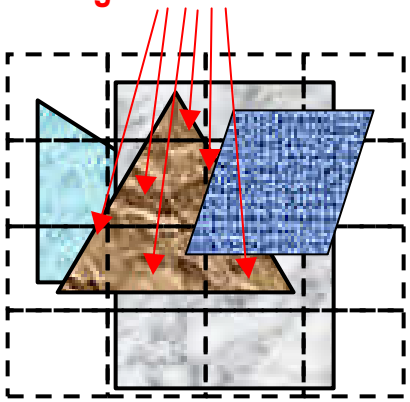


# Tile-Based Rendering Pipeline

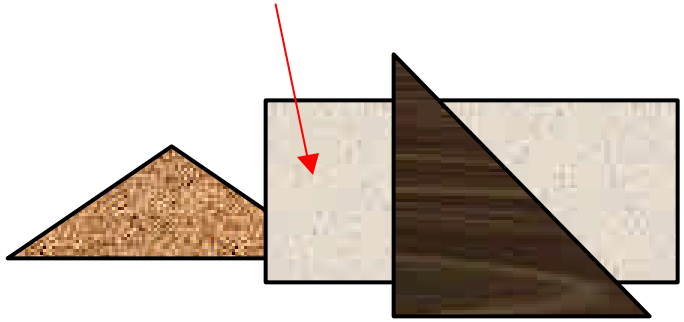
## *Problems with with real-world situations*

- **Sorting and front-to-back rendering are not always possible**
  - Back-to-front rendering is often required for blending and transparency effects
  - Some applications don't use Z-buffering
- **Finite Bin-store means external Z-buffer memory cannot be eliminated**
  - The bin-store is needed to store all polygons touching each tile on the screen
  - It is possible the bin-store overflows when scene complexity increases
  - In this case need to process all the bins and then process more polygons
  - An external Z buffer is required in order to enable to store intermediate Z results
  - The internal Z-buffer tile be flushed and loaded for each tile if bin-store overflows
- **TBR means texture access efficiency is reduced**

**In a TBR architecture, a primitive can be processed over multiple tiles – potentially forcing texture to be loaded multiple times**



**In a standard architecture, primitives are processed in order, meaning that textures are typically loaded and cached once**



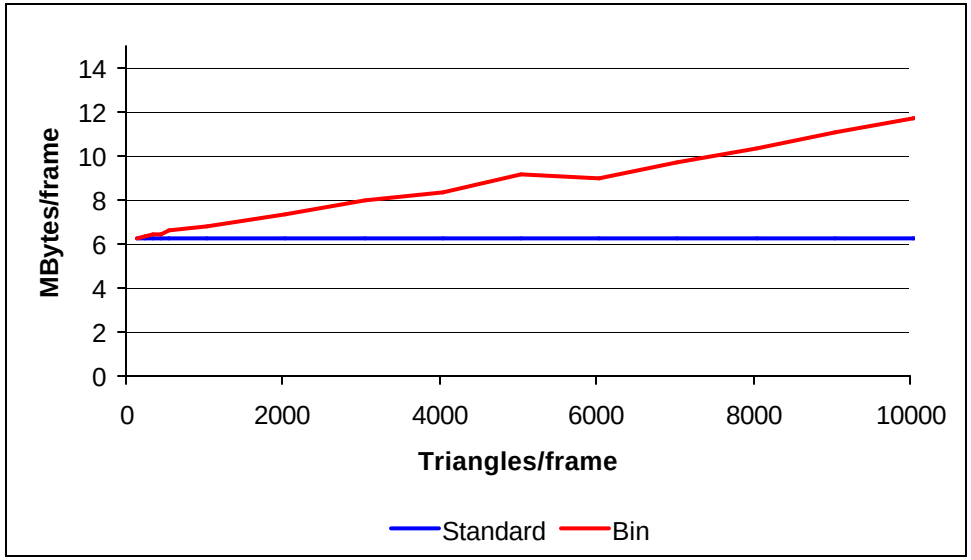
# Bandwidth Requirements Example

## *Assumptions*

- **Screen resolution**
  - 320 x 240
- **Vertex format**
  - 52 bytes per vertex (position, normal, diffuse, specular, fog, 2 sets of texture coordinates)
- **Texture cache hit rate**
  - 85% for traditional pipeline (takes advantage of per polygon texture coherency)
  - 50% for RBR pipeline (depends on cache size)
- **Extra Bin Store memory**
  - 600Kbytes
- **Tile size**
  - 16 x 16 or 32 x 32
    - Trade-off between texture cache hit rate and amount of on-chip memory
- **Texture compression**
  - 1:8

3Dlabs

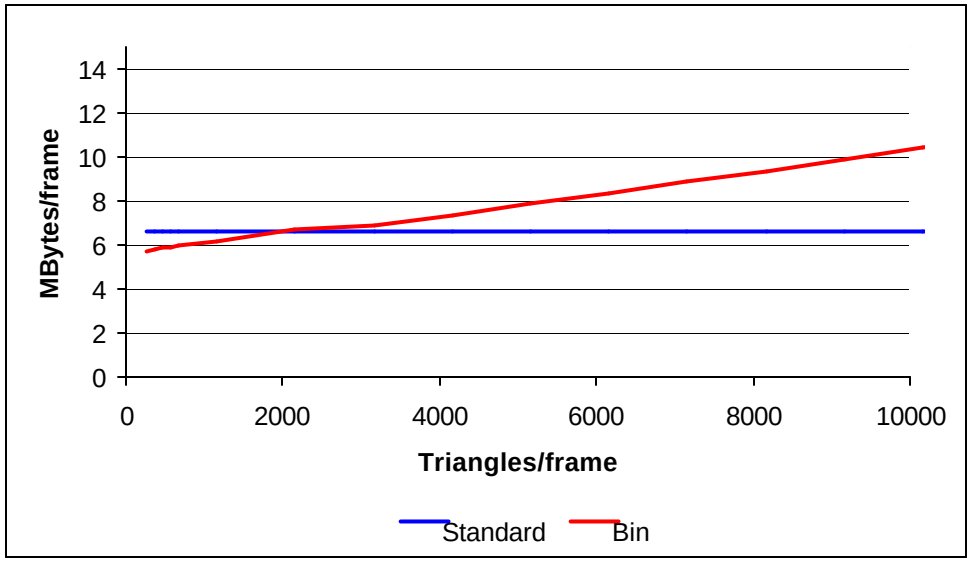
# Bandwidth Requirements Example




**16 x 16 tiles**

**TBR takes more memory and only gives a memory bandwidth advantage for simple scenes.**
**How can standard architectures gain some of the small scene advantages of TBR?**

**32 x 32 tiles**



# Enhancing The Traditional Pipeline

*Absorbing the advantages of tile-based rendering*

- **Beware simplistic overwrite elimination benefit calculations**
  - In a traditional pipeline, when each pixel is covered by 3 polygons, each pixel is touched 1.83 times on average, not 3 times. When each pixel is covered by 4 polygons, each pixel is touched 2.1 times on average, not 4 times
- **Hierarchical Z-buffering**
  - Storing a Z value for a region of the screen and fast rejecting primitives against it
  - Significantly reduces the number of per-pixel Z rejections for hidden polygons
- **Two-pass Z-buffers**
  - If performing very complex shading calculations then calculate just the Z map on first path
  - On second pass reject against the Z before spending time on shading calculations
- **Object-level culling and bounding box hierarchy culling**
  - Eliminate depth complexity by culling objects before they hit the graphics pipeline at all

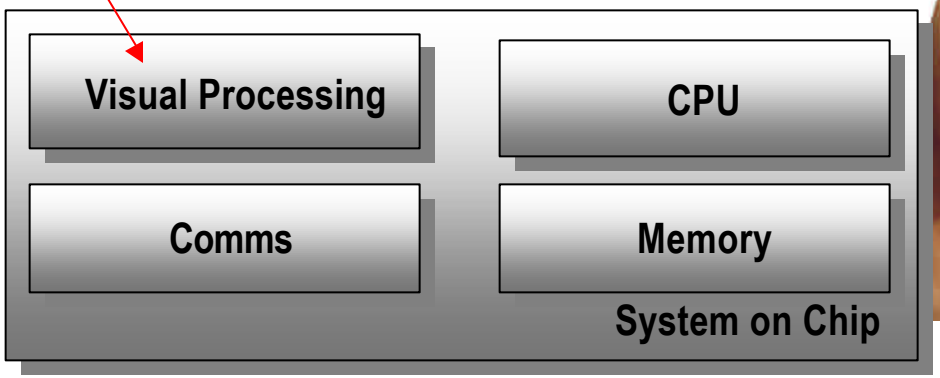
3Dlabs

# Appliances need System on Chip

## *Enabling a new class of computing platforms*

- **IP cores for SoC is a fundamental enabler for advanced appliances**
  - Absorb all system functionality into one chip
  - CPU, visual processing, communications, memory
- **All 3Dlabs chips are now also available as cores**
  - Enabling SoC integration while preserving software investment
- **Enabling re-use and integration of diverse IP is still challenging**
  - The graphics vendors have some of the most experienced physical design teams

**3Dlabs supplying Graphics and Video Cores**




# 3Dlabs IP Cores

*A range of power and performance options*

- **Enabling a range of price/performance options**
  - Extensive software compatibility between classes
- **V1/V2 generation cores are available now and have been instanced**
  - Great for PDAs, TabletPCs, low-end STBs, DTVs, Automotive Display Systems
- **V3 generation core will be available in Q1, instanced in Q3**
  - Great for high-end STBs, games consoles
- **Video codecs within 6 months**
  - Multi-stream, multi-format encoders and decoders as optimized silicon cores

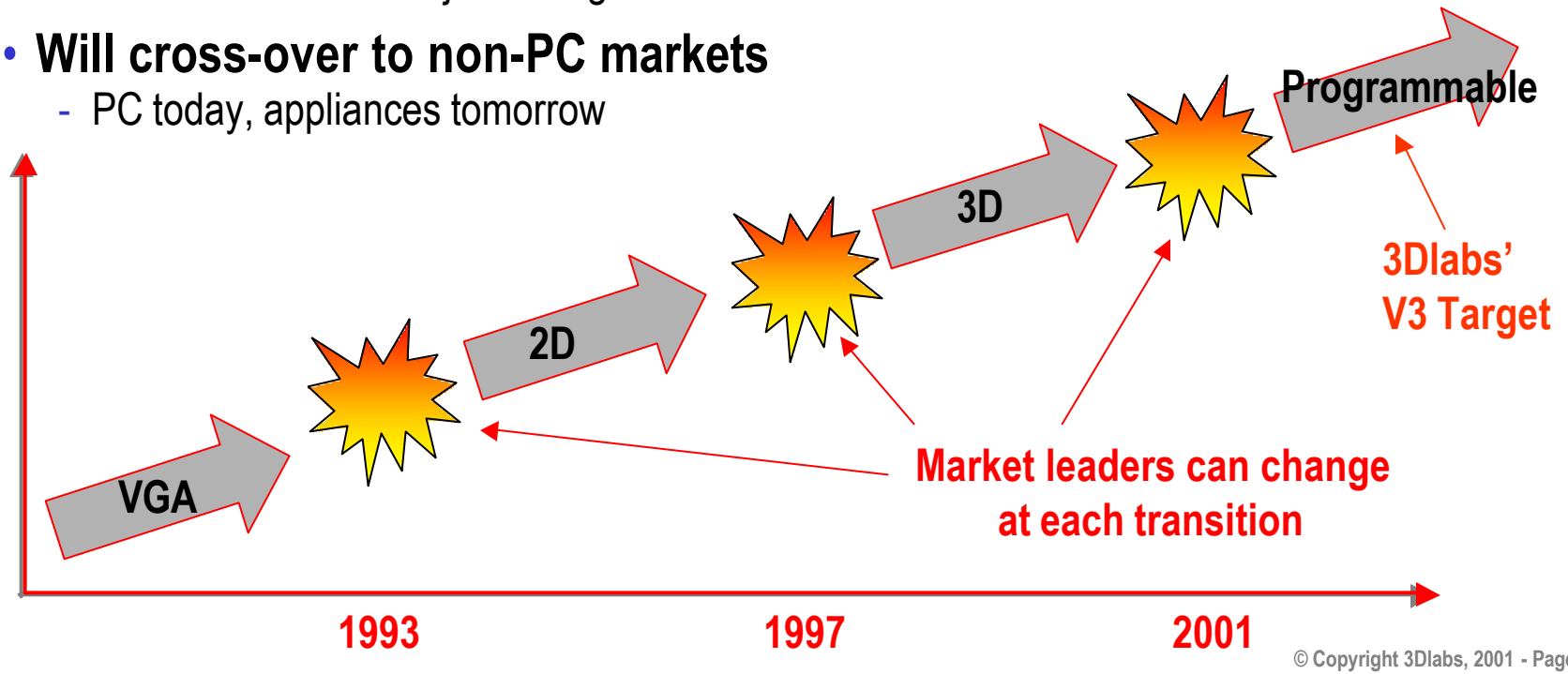
| | | |
|---|---|---|
| **V1 Class** | **Gamma 1, Permedia 2** | **Low-cost and power >2x2mm$^2$ in 0.15** |
| **V2 Class** | **Gamma 2/3, GLINT R3/4** | **Medium Power >5x5mm$^2$ in 0.15** |
| **V3 Class** | **Advanced Programmable Visual Architectures** | **Higher Power >7x7mm$^2$ in 0.15** |

# Leveraging IP on the PC

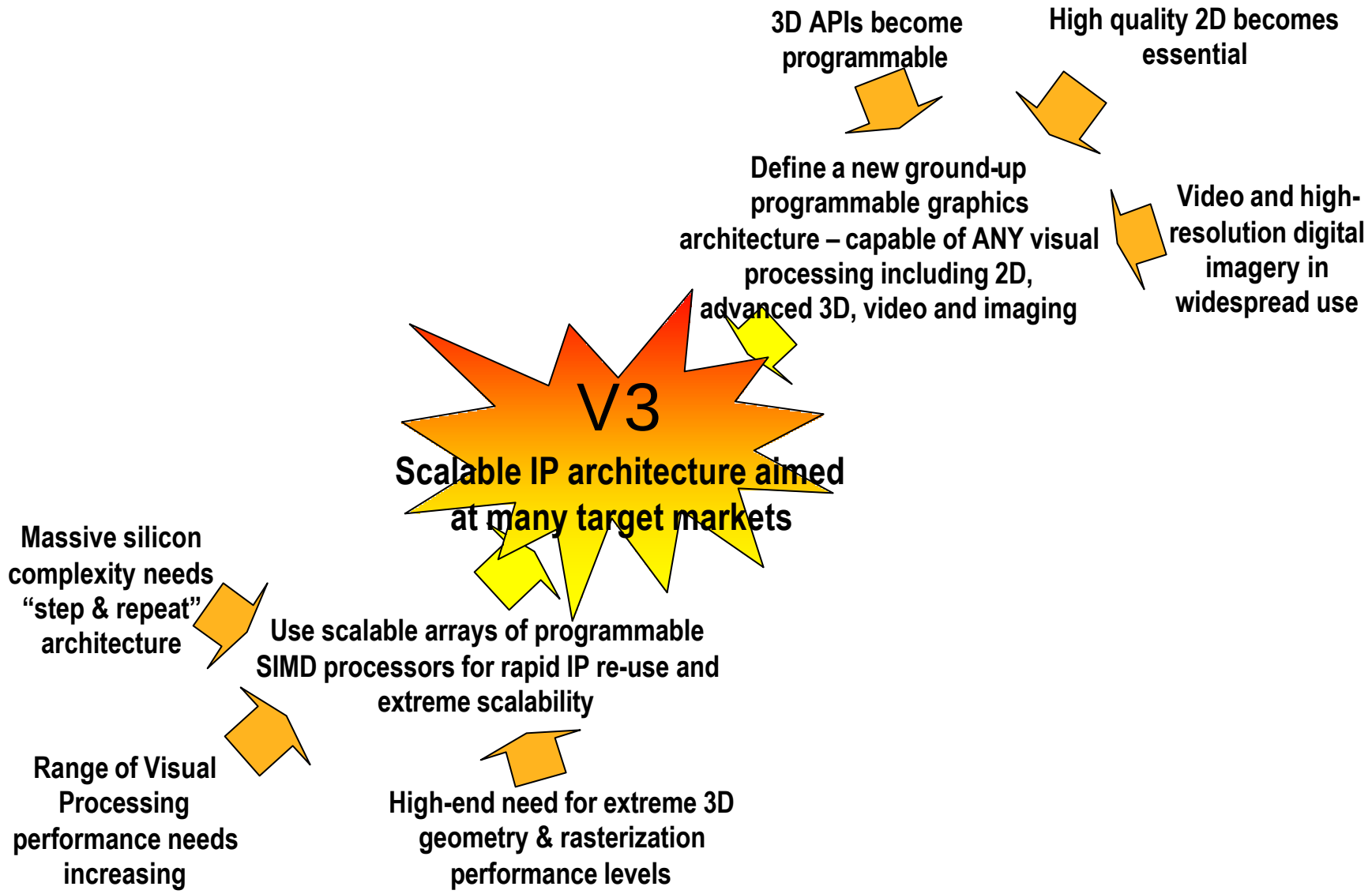
## *Exploiting the paradigm shift*

- **D3D is evolving at an rapid rate**
  - Allows IHVs to extend HW functionality
- **Moving from fixed function to programmable pipeline**
  - Will no longer be an API, more a form of expression
  - Next step towards hardware-accelerated “Toy Story” quality graphics
- **The graphics market leaders changed in the 2D to 3D transition**
  - This transition will be just as significant
- **Will cross-over to non-PC markets**
  - PC today, appliances tomorrow



# The Need for a New Architecture

## *Defining the first Visual Processing Unit - VPU*

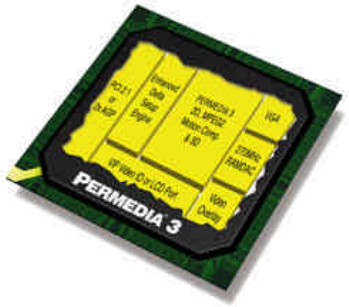


# Licensing Successes

*Reduced time-to-market, low-risk, right-first time*

- **TI Permedia**
  - 3Dlabs design originally fabbed at IBM – 2M transistors
  - TI licensed the Permedia soft cores from 3Dlabs
  - TI instanced a faster chip in 3 months
  - Project included 100% re-synthesis, 100% re-verification,
  - The chip was right first time
  - TI engineers involved – 6
  - 3Dlabs engineers involved – 2

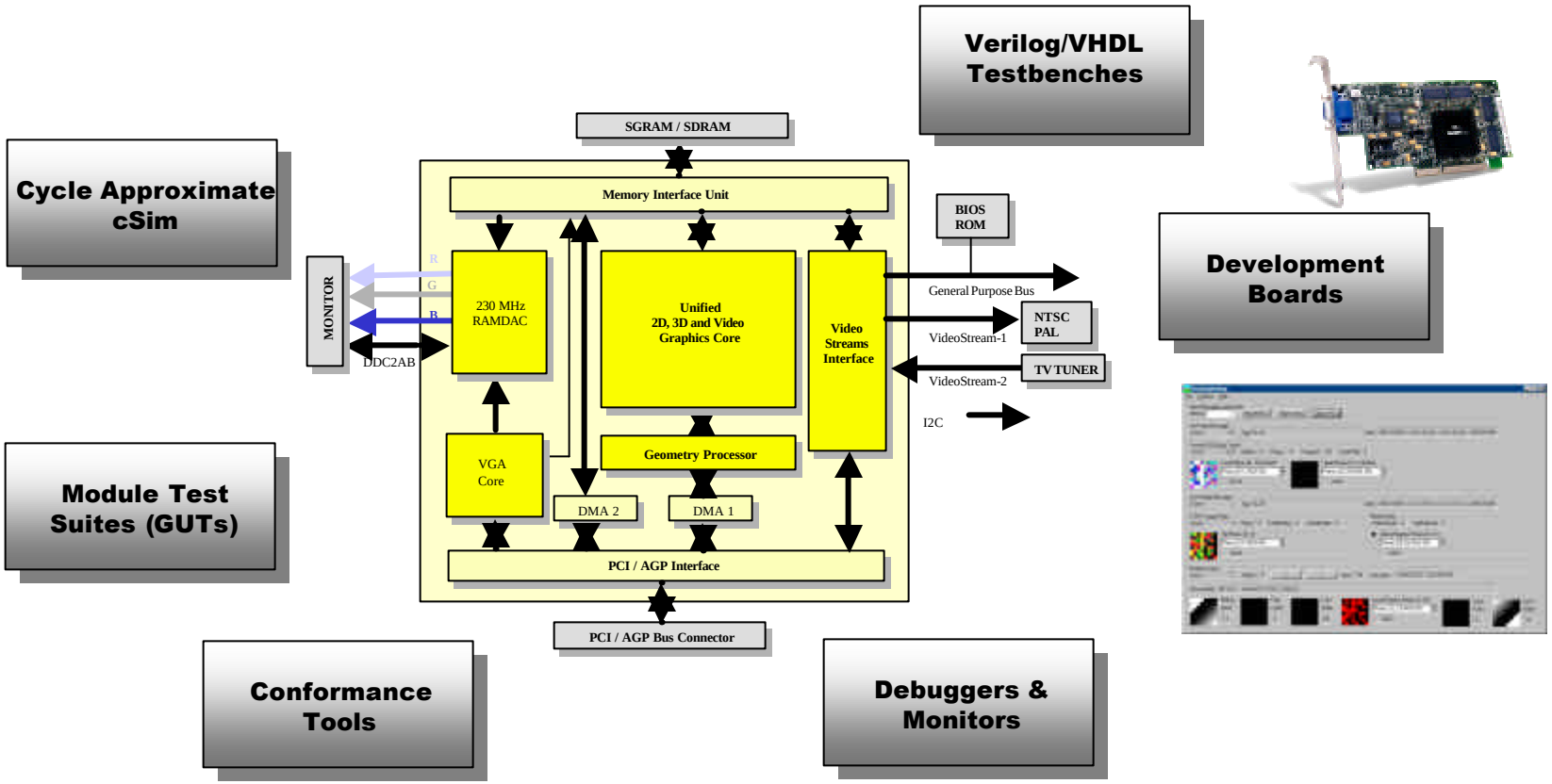

- **Appian AGX**
  - 3Dlabs design originally fabbed at IBM – 8M transistors
  - Appian licensed Permedia 3 soft and hard cores from 3Dlabs
  - Appian added a second RAMDAC, re-synthesized memory interface at a higher speed
  - Appian re-instanced the chip at IBM in 6 months – their first ever chip design
  - The chip was right first time
  - Appian engineers involved - 6
  - 3Dlabs engineers involved – 1



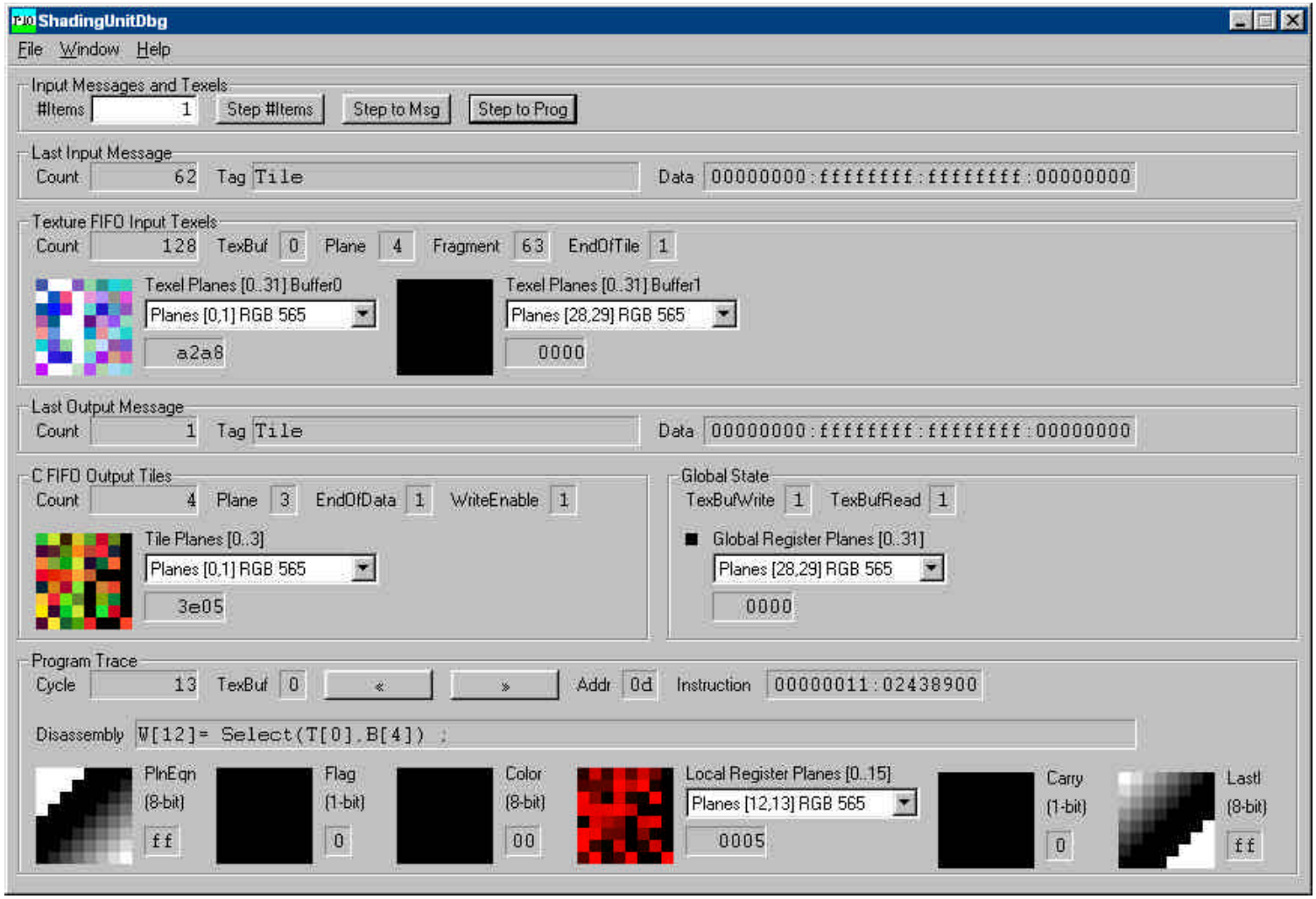
# IP Development solutions

## *Productivity through tools and methods*

- **Development and debugging tools allow rapid prototyping**
  - Use standard PC and Sun platforms
  - Allow module-level and full chip debugging
  - Cycle approximate cSim allows software development and verification
  - No hardware emulators are necessary



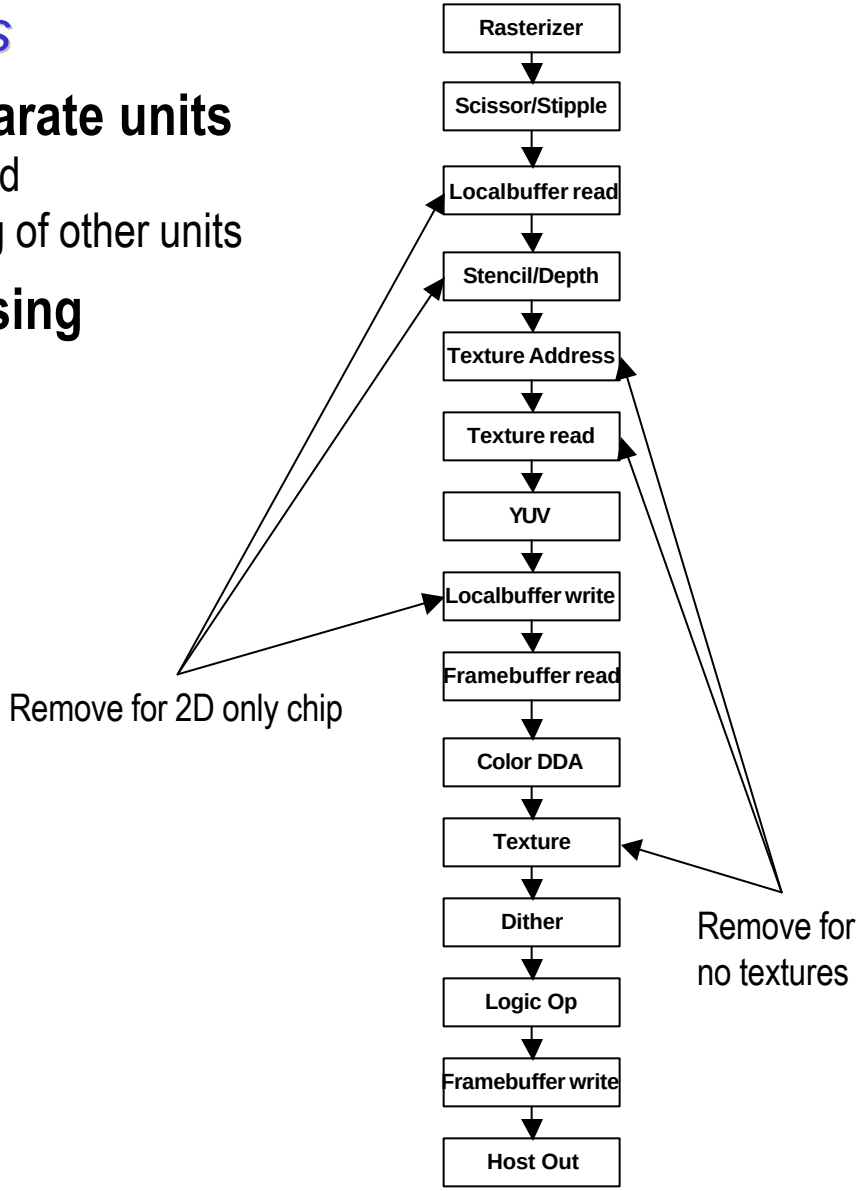
# Visual Debugger



# Modularity of Function

*Tune for cost-effectiveness*

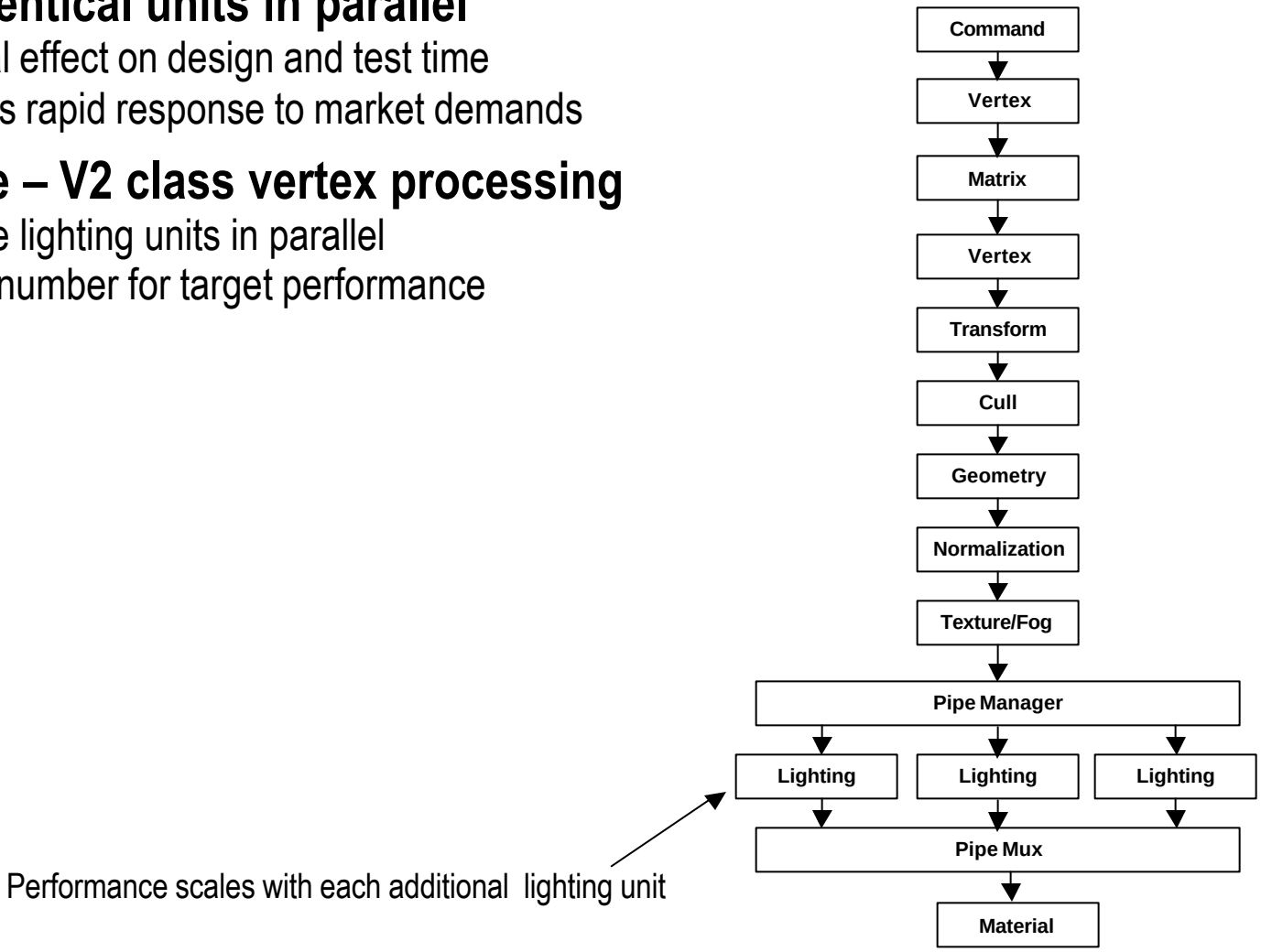
- **Functions are performed by separate units**
  - Units not needed can be easily removed
  - Does not affect functionality and testing of other units
- **Example - V1 class pixel processing**
  - If target application is 2D only
    - Remove depth buffer units
  - If target application uses no textures
    - Remove texture units



# Modularity of Performance

*Tune for the right level of interaction*

- **Gang identical units in parallel**
  - Minimal effect on design and test time
  - Enables rapid response to market demands
- **Example – V2 class vertex processing**
  - Multiple lighting units in parallel
  - Select number for target performance



# Summary

## *3Dlabs - taking the lead in embedded visual processing*

- **3Dlabs is fully committed to the embedded market**
  - We are deploying increasing resources to support our embedded customers
- **We are offering chips and IP cores**
  - Enabling advanced visual processing in a wide range of platforms
- **We are spearheading a visual processing API open standard initiative**
  - Enabling overall market growth
- **We are generating a new class of visual processing IP**
  - The first true Visual Processing Units
  - VPUs - processing graphics, images and video

# Tile-Based Rendering

*Is it the right way to implement embedded graphics?*

- **Tile-based has some potential advantagesRBR yields lower bandwidth requirements for a given fill rate**
  - On-chip Z-buffering reduces external memory bandwidth requirements
    - No Z reads/writes from/to external memory
  - No overdraw when deferred shading is implemented
    - Reduces texture reads
      - The fragments that will not be visible in the final scene are not rendered to the frame buffer
- **Deferred shading yields a lower gate count for a given fill rate**
  - For a given gate count, deferred shading increases the visible pixel fill rate by a factor equal to the depth complexity
    - DC = average number of polygons that cover a pixel
    - Primitive based pipeline: visible pixel fill rate = fragment rate / DC
    - Deferred shading pipeline: visible pixel fill rate = fragment rate
- **Let's see …**

# Traditional Pipeline

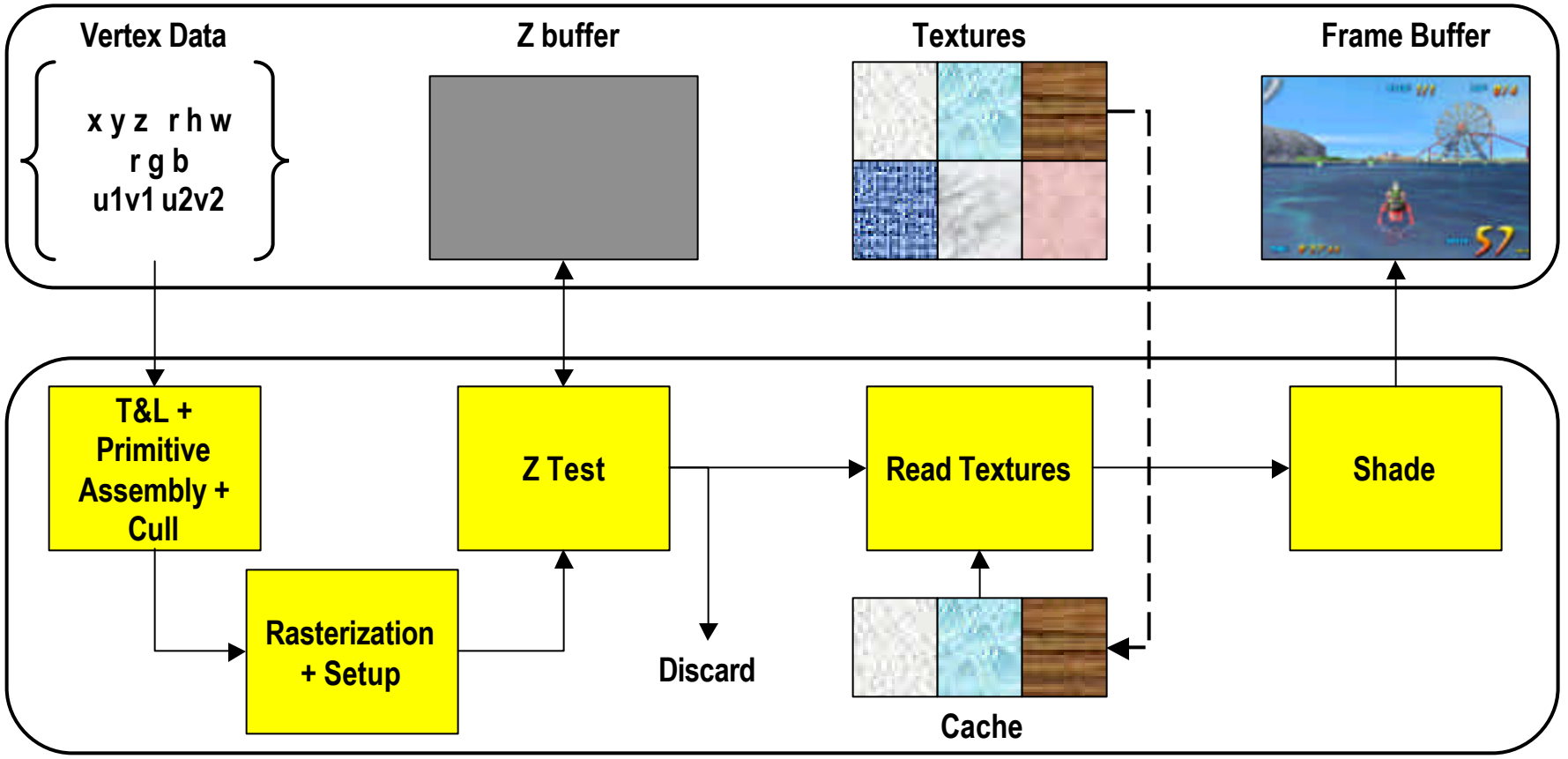
## Advantages

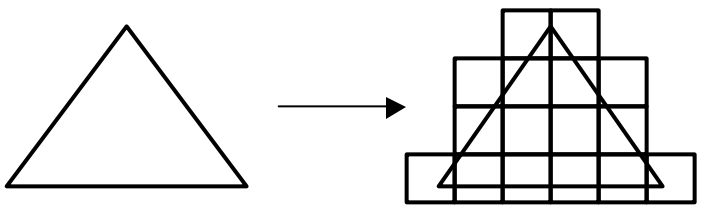
- **"Natural" rendering flow**
  - Render a database the same way it is defined and traversed: scene✍object✍polygon
- **Can handle any number of polygons**
  - Polygons are processed sequentially
  - Doesn't use any intermediate geometry buffer
- **Low latency**
  - Same reasons as above
- **Simple hidden surface removal**
  - Simple algorithm, simple circuitry
    - For each fragment: 1) read stored Z value, 2) compare with current Z value, 3) write or discard current Z value
- **High texture cache efficiency**
  - Textures are associated to polygons, not to screen coordinates
- **Small amount of data required for setup and rasterization**
  - Vertex data, interpolators

## Drawbacks

- **Overdraw**
  - Some polygons are shaded and written to the frame buffer, only to be overwritten later
- **Z-buffering external memory requirements**

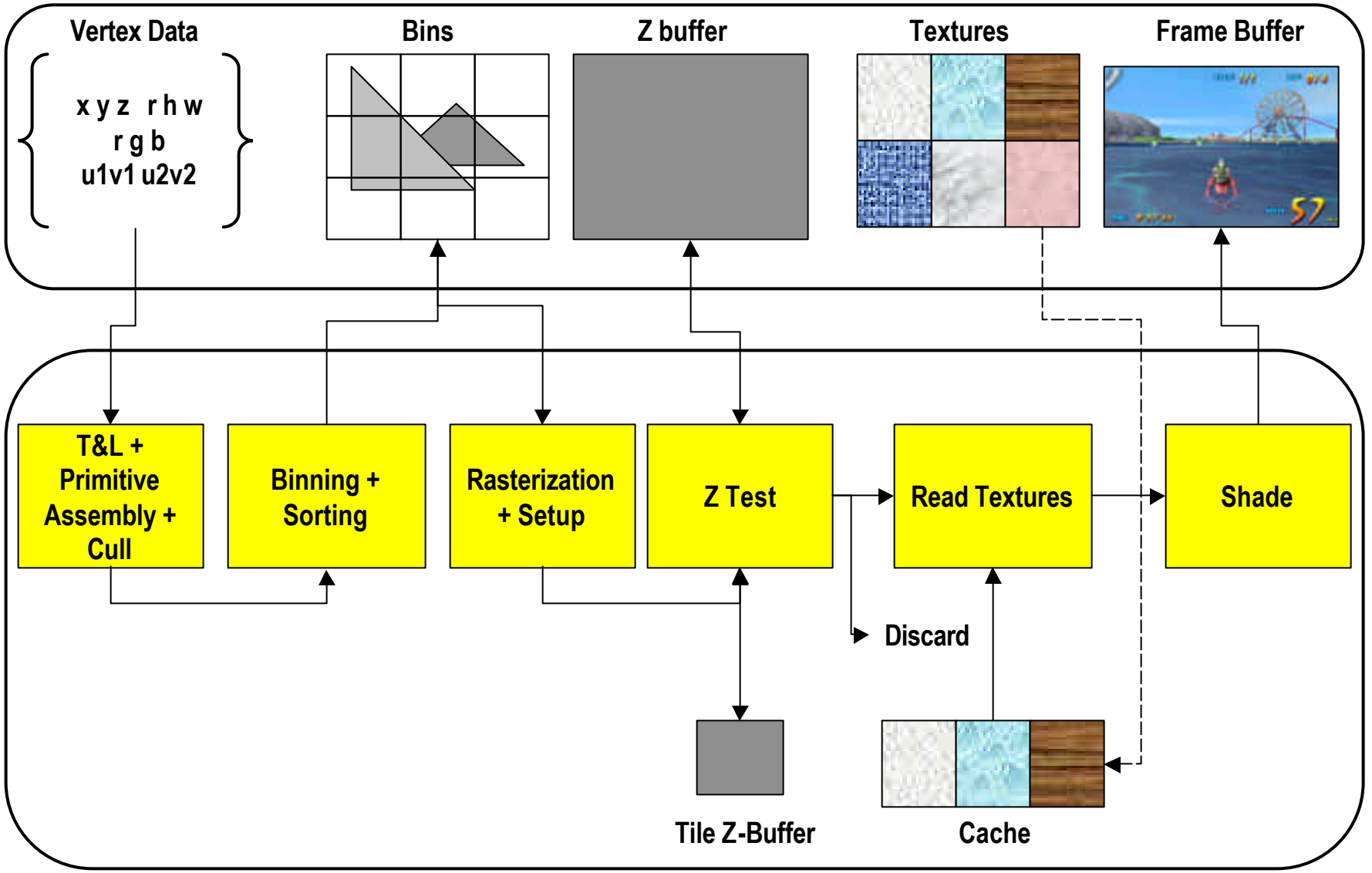
# Traditional Polygon-Based Pipeline



# Region-Based Rendering Pipeline

*Example*



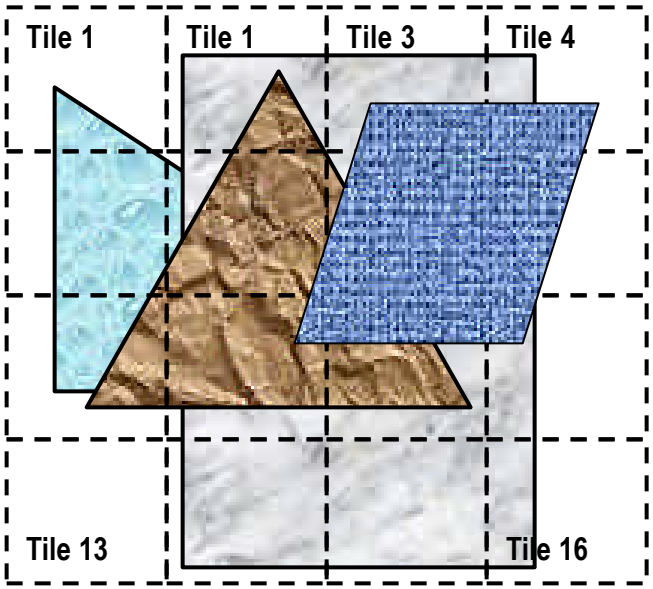
# Region-Based Rendering Pipeline

## *Explanation*

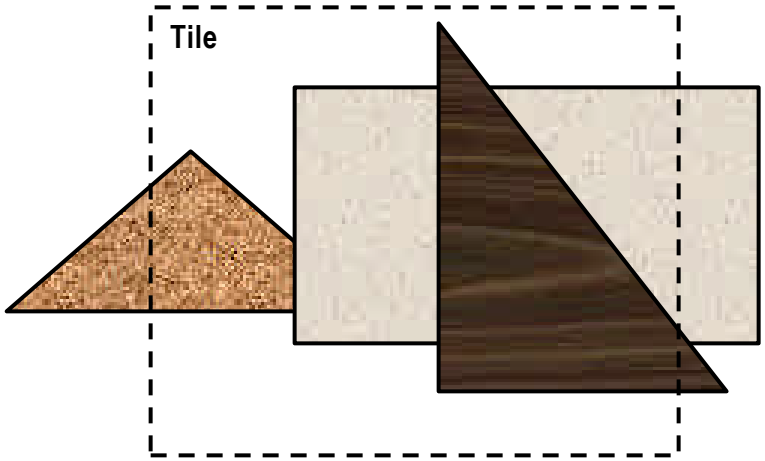
- **T&L**
  - Transform and light vertices in vertex buffer
  - Primitive assembly
- **Binning**
  - One bin for each tile
  - The bin size is not unlimited
  - Determine all the tiles that cover each polygon
  - Fill bins with clipped polygons
  - Flush the bins when they're full
  - Repeat
- **Sorting (optional)**
  - Perform depth sorting for each bin to reduce overdraw
  - Alternative 2-pass implementation: compute the full Z-buffer for the tile, then render.
- **Z test and Z buffers**
  - A Z buffer is required in order to be able to have multiple passes through the bins
    - This is required in order to be able to handle any number of polygons with a finite amount of memory
  - On-chip Z buffer for current tile
  - Off-chip full screen Z-buffers for all tiles
    - The Z buffers for all the tiles/bins must be stored in order to be able to handle multiple passes through the bins
- **Per fragment operations**
  - Use front to back rendering or scan-line rendering inside each tile to limit overdraw
  - Region-based rendering does not take full advantage of per-polygon texture coherency, especially when deferred shading is used ✍ Low texture cache efficiency

3Dlabs

# Texture Reads With RBR




**Each texture is accessed several times, even if it is used on a single polygon**
as long as it is used in several tiles




**Overall, texture swapping is much more frequent than in the traditional polygon-based pipeline**

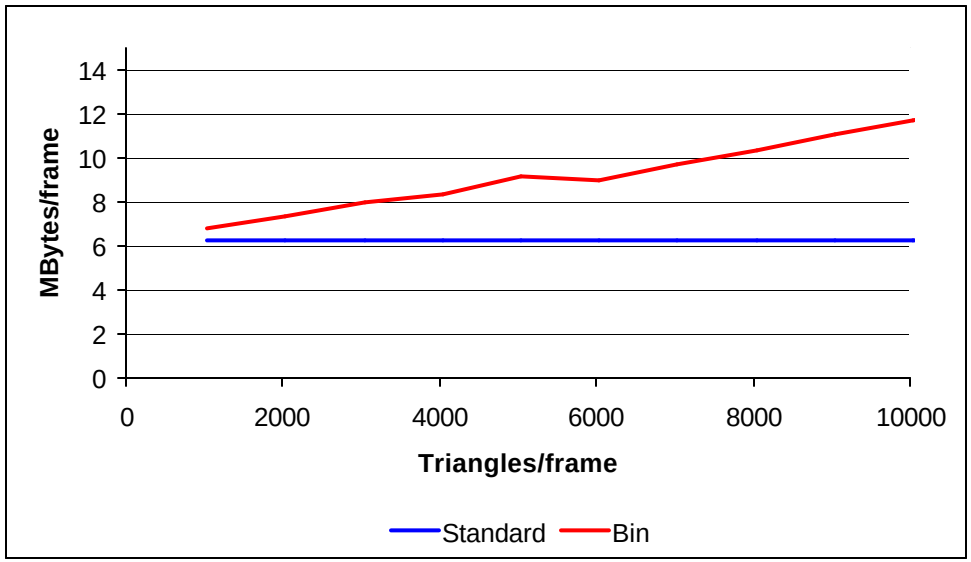
# Bandwidth Requirements Example

## *Assumptions*

- **Screen resolution**
  - 320 x 240
- **Vertex format**
  - 52 bytes per vertex (position, normal, diffuse, specular, fog, 2 sets of texture coordinates)
- **Texture cache hit rate**
  - 85% for traditional pipeline (takes advantage of per polygon texture coherency)
  - .5 for RBR pipeline
    - Uninformed estimate based on the fact that the textures are not accessed in any particular order
    - But depends on cache size of course
- **Number of times the bins are filled**
  - 3
    - Depends on scene complexity and available memory
    - Requires 1MB memory + texture memory with RBR
    - Requires 0.4MB memory + texture memory with polygon based rendering
- **Tile size**
  - 16 x 16 or 32 x 32
    - Trade-off between texture cache hit rate and amount of on-chip memory
- **Texture compression**
  - 1:8

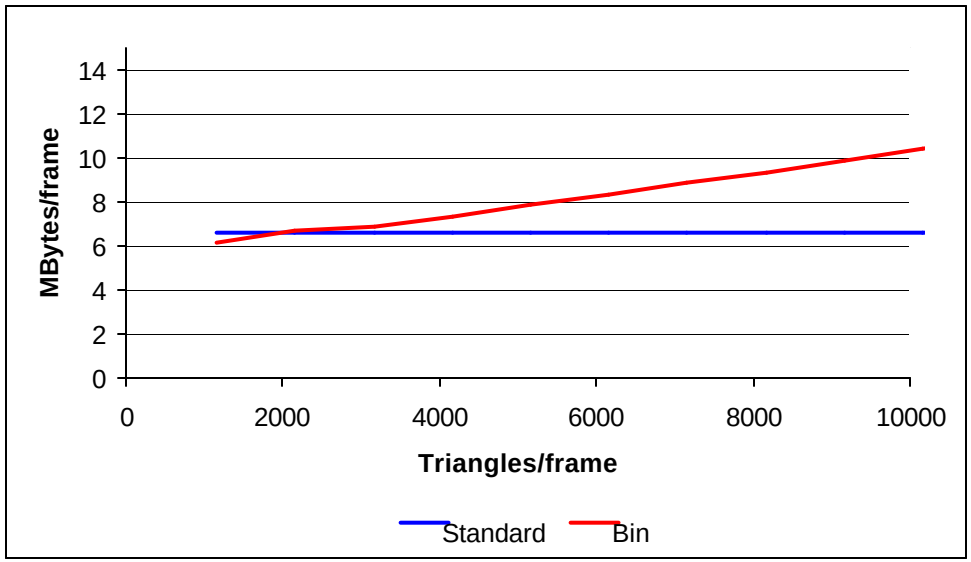
3Dlabs

# Bandwidth Requirements Example




**16 x 16 tiles**

**32 x 32 tiles**



# RBR and Industry-Standard APIs

## *Still Some Issues*

- **Sorting and front-to-back rendering are not always possible**
  - Back-to-front rendering is often required for blending and transparency effects
- **Some applications don't work with Z-buffering**
  - Applications that use back-to-front rendering
  - But Z buffering must be enabled in order to be able to handle any number of polygons

# Region-Based Rendering

## *A Few Additional Factors To Consider*

- **The cost of binning**
  - Gates for binning engine
  - External memory storage space
  - External memory bandwidth
- **The cost of sorting and deferred shading**
  - Gates for sorting engine, or 2-pass rendering (Z-buffer computed during the first pass, visible polygons are rendered in the second pass)
  - External memory bandwidth
- **The cost of on-chip tile and Z buffers**
  - Gates for on-chip memory Z-buffer and tile buffers
- **Low texture cache efficiency**
  - Traditional primitive-based pipelines take advantage of per-polygon texture coherency, region-based rendering doesn't
- **The benefit of deferred shading is not as high as the marketing brochures claim**
  - Sorting and deferred shading do not eliminate overdraw
    - Overdraw is eliminated only when each bin is filled only once and scanline rendering is used
  - Traditional Z-buffering is not as inefficient as some people would have you believe
    - In a traditional pipeline, when each pixel is covered by 3 polygons, each pixel is touched 1.83 times on average, not 3 times. When each pixel is covered by 4 polygons, each pixel is touched 2.1 times on average, not 4 times.
- **RBR introduces another potential bottleneck in the pipeline**
  - Sorting and binning
  - Increasing 3D performance is all about removing bottlenecks
  - Have you actually seen a chip that performs T&L, binning, sorting, on-chip Z-buffering, and that can sustain 10M+ polygons per second? If so, how big was it?

3Dlabs

# The Traditional Pipeline

## *Getting More Efficient Every Year*

- **Back-end bandwidth requirements can be reduced without increasing front-end complexity and bandwidth requirements**
  - Hierarchical Z-buffering schemes
    - To reduce bandwidth requirements for Z reads and writes
  - Texture compression
    - To reduce bandwidth requirements for texture reads
- **Overdraw can also be reduced in the traditional pipeline**
  - Applications can (and will) do deferred rendering in the traditional pipeline
    - The two-pass Z-buffer-based technique can be implemented by the app
  - Overdraw can be reduced by reducing the depth complexity of the scene
    - Object-level culling and bounding box hierarchy culling can (and will) be used to reduce depth complexity in complex scenes
- **As these techniques are implemented in future chips and apps, will the trade-offs associated with RBR still be interesting?**
  - Stay tuned!

3Dlabs